Available online at www.sciencedirect.com

ScienceDirect

Procedia Social and Behavioral Sciences 17 (2011) 341–363

Procedia Social and Behavioral Sciences

19th International Symposium on Transportation and Traffic Theory

A Tractable Class of Algorithms for Reliable Routing in Stochastic Networks

S. Samaranayake[a,*], S. Blandin[a], A. Bayen[b]

[a]*PhD student, Systems Engineering, University of California Berkeley, 621 Sutardja Dai Hall, Berkeley, CA 94720-1764, USA.*
[b]*Associate Professor, Systems Engineering, University of California Berkeley, 642 Sutardja Dai Hall, Berkeley, CA 94720-1764, USA.*

Abstract

The goal of this article is to provide the theoretical basis for enabling tractable solutions to the "arriving on time" problem and enabling its use in real-time mobile phone applications. Optimal routing in transportation networks with highly varying traffic conditions is a challenging problem due to the stochastic nature of travel-times on links of the network. The definition of optimality criteria and the design of solution methods must account for the random nature of the travel-time on each link. Most common routing algorithms consider the expected value of link travel-time as a sufficient statistic for the problem and produce least expected travel-time paths without consideration of travel-time variability. However, in numerous practical settings the reliability of the route is also an important decision factor. In this article, we consider the following optimality criterion: *maximizing the probability of arriving on time at a destination given a departure time and a time budget*. We present an efficient algorithm for finding an optimal routing policy with a well bounded computational complexity, improving on an existing solution that takes an unbounded number of iterations to converge to the optimal solution. A routing policy is an adaptive algorithm that determines the optimal solution based on en route travel-times and therefore provides better reliability guarantees than an a-priori solution. Novel speed-up techniques to efficiently compute the adaptive optimal strategy and methods to prune the search space of the problem are also investigated. Finally, an extension of this algorithm which allows for both time-varying traffic conditions and spatio-temporal correlations of link travel-time distributions is presented. The dramatic runtime improvements provided by the algorithm are demonstrated for practical scenarios in California.

Keywords: stochastic routing, adaptive routing, dynamic programming, traffic information systems, real-time algorithms

1. Introduction

The design of optimal routing systems for operational scenarios in large scale transportation networks is a challenging problem due to the variability of realized link travel-times. As a result of this variability, the total travel-time on each link (and therefore the entire route) is a random variable with an associated probability distribution. Depending on the user preferences, an optimal route in this setting might need to consider notions of both the travel-time (expected value) and travel-time reliability (variance). This is a multi-criterion optimization problem that is in general

*Corresponding author
Email addresses: samitha@berkeley.edu (S. Samaranayake), blandin@berkeley.edu (S. Blandin), bayen@berkeley.edu (A. Bayen)

doi:10.1016/j.sbspro.2011.04.521

hard to solve. The most commonly used formulation of optimality is to consider the route with the *least expected time* (LET) as defined by Loui (1983). The LET problem has been well researched and many efficient algorithms exist for different variants of the problem; for example Fu and Rilett (1998); Miller-Hooks and Mahmassani (2000); Waller and Ziliaskopoulos (2002). When the link weights are independent and time-invariant distributions, the LET problem can be reduced to the standard deterministic shortest path problem by setting each link weight to its expected value. The solution to this problem can be computed efficiently using the Dijkstra (1959) algorithm. Hall (1986) shows that Dijkstra's algorithm does not provide an optimal solution when the link weights are time-varying. If the network satisfies the first-in first-out (FIFO) condition defined by Astarita (1996), the problem can be solved using dynamic programming using time-dependent weights with the original graph, see Dean (2004). Waller and Ziliaskopoulos (2002) consider the time-invariant LET problem with correlated link travel-times.

However, there are several settings in which the LET solution is not adequate, since it does not take into account the variance of travel-time distributions and gives no reliability guarantees. The optimal path defined by the LET solution can be unreliable and can result in highly variable realizations of travel-time. In numerous cases, travelers have hard deadlines or are willing to sacrifice travel-time to take a more reliable route. In commercial routing, there are delivery guarantees that need to be met and perishables that need to be delivered within a fixed amount of time. Frank (1969) presents a very natural definition of a reliable optimal path, as the path that maximizes the probability of realizing a travel-time that is less than a given constant. However, the formulation given by Frank requires enumerating all possible paths and therefore is not tractable for practical problems.

A formulation of the problem using stochastic optimal control (see Bertsekas (2005)), which combines static information about the network structure with real-time information about actual travel-times, results in an adaptive solution that is an optimal policy as opposed to an optimal path. An optimal policy generates a node-based decision rule that defines the optimal path from a given node to the destination conditioned on the realized travel-time. It is clear that such an adaptive policy should do better than a static a-priori solution.

Fan and Nie (2006) consider Frank's formulation of maximizing the probability of realizing a given travel-time, also known as the *stochastic on time arrival* (SOTA) problem, and formulate it as a stochastic dynamic programming problem. The proposed dynamic program is solved using a standard *successive approximation* (SA) algorithm. In an acyclic network, the SA algorithm converges in a number of steps no greater than the maximum number of links in the optimal path. However, in a realistic network an optimal adaptive strategy may contain loops. Moreover, there is no finite bound on the maximum number of links the optimal path can contain, as explained by Fan and Nie (2006). Therefore, the number of steps required for the algorithm to converge is unbounded. As an alternative, Nie and Fan (2006) propose a discrete approximation algorithm for the SOTA problem that converges in a finite number of steps and runs in pseudo-polynomial time.

In this article, we present a number of theoretical and numerical results that improve the tractability of the SOTA problem over existing methods. We first solve the unbounded convergence problem, by developing a new algorithm that gives an exact solution to the SOTA problem and has a provable convergence bound. As with Fan and Nie (2006), this algorithm requires computing a continuous-time convolution product, which is one of the challenges of the method. In general, this convolution cannot be solved analytically when routing in arbitrary networks, and therefore a discrete approximation scheme is required. By exploiting the structure of our algorithm, we are able to solve the convolution more efficiently than the standard (brute force) discrete time approximation algorithm used in Nie and Fan (2006) and obtain a faster computation time. We show that the order in which the nodes of the graph are considered greatly impacts the running time of our proposed solution and present an optimal ordering algorithm that minimizes the computation time.

In addition, we present an analysis of the conditions under which our framework can be extended to handle time-varying travel-time distributions and show that these conditions are satisfied in the commonly used travel-time models for road networks. We also consider the problem of correlated travel-time distributions. Finally, we present a network pruning scheme that reduces the search space of the algorithm and thus also improves its computational efficiency. cr Our goal is to provide the theoretical basis for a tractable implementation of adaptive routing with reliability guarantees in an operational setting. One specific application of interest is to enable adaptive routing on mobile phones using real-time traffic data. Experimental results are provided for San Francisco Bay Area highway and arterial networks using the *Mobile Millennium* (2008) traffic information system.

The rest of the article is organized as follows. In Section 2, we define the *stochastic on time arrival* (SOTA) problem and discuss its classical solution method. In Section 3, we present a new SOTA algorithm, we prove its

convergence properties and discuss how the algorithm can be used with both time-varying and correlated travel-time distributions. In Section 4, we present an efficient numerical method to approximate convolution integrals using the Fast Fourier Transform (FFT) and an optimal update algorithm. Experimental results are given in Section 5. Finally, we present our conclusions in Section 6.

2. The Stochastic On-time Arrival (SOTA) Problem

We consider a directed network $G(N, A)$ with $|N| = n$ nodes and $|A| = m$ links. The weight of each link $(i, j) \in A$ is a random variable with probability density function $p_{ij}(\cdot)$ that represents the travel-time on link (i, j). Given a time budget T, an optimal routing strategy is defined to be a policy that maximizes the probability of arriving at a destination node s within time T. A routing policy is an adaptive solution that determines the optimal path at each node (intersection in the road network) based on the travel-time realized to that point. This is in contrast to a-priori solutions that determine the entire path prior to departure. Given a node $i \in N$ and a time budget t, $u_i(t)$ denotes the probability of reaching node s from node i in less than time t when following the optimal policy. At each node i, the traveler should pick the link (i, j) that maximizes the probability of arriving on time at the destination. If j is the next node being visited after node i and ω is the time spent on link (i, j), the traveler starting at node i with a time budget t has a time budget of $t - \omega$ to travel from j to the destination, as described in equation (1)[1].

Definition 1. *The optimal routing policy for the SOTA problem can be formulated as follows:*

$$\begin{aligned} u_i(t) &= \max_j \int_0^t p_{ij}(\omega)\, u_j(t-\omega)\, d\omega \\ &\qquad \forall i \in N,\ \ i \neq s,\ \ (i,j) \in A,\ \ 0 \leq t \leq T \\ u_s(t) &= 1 \quad 0 \leq t \leq T \end{aligned} \tag{1}$$

where $p_{ij}(\cdot)$ is the travel-time distribution on link (i, j).

The functions $p_{ij}(\cdot)$ are assumed to be known and can for example be obtained using historical data or real-time traffic information.

Fan and Nie (2006) present the *successive approximations* (SA) algorithm described in Algorithm 1, which solves the system of equations (1) and gives an optimal routing policy.

At each iteration k, $u_i^k(t)$ gives the probability of reaching the destination from node i within a time budget t, using a path with no more than k links, under the optimal policy. The approximation error monotonically decreases with k and the solution eventually reaches an optimal value when k is equal to the number of links in the optimal path. A formal proof of the convergence is given in Section 3 of Fan and Nie (2006) using the bounded monotone convergence theorem. However, since an optimal routing policy in a stochastic network can have loops (see Example 1), the number of iterations required to attain convergence is not known a-priori.

Example 1. *Figure 1 shows a simple network in which an optimal path can contain a loop. Consider finding the optimal route from node a to node c with a budget of* 4 *time units. There are two choices at the origin, of which it is clear that link* (a, b) *gives the highest probability of reaching the destination on time, since there is a* 0.9 *probability of the travel-time on* (a, b) *being 1 time unit, which leads to a total travel-time of* 4*. However, assume that the realized travel-time on link* (a, b) *is actually 2 time units, which can happen with probability* 0.1*. In this case, taking the path* $\{(a, b), (b, c)\}$ *results in zero probability of reaching the destination on time. The optimal path is therefore* $\{(a, b), (b, a), (a, c)\}$*, which is the only path that has a non zero probability of reaching the destination on time. This optimal path contains a loop. An a-priori algorithm such as the least expected time (LET) algorithm will always route on path* $\{(a, b), (b, c)\}$ *regardless of the realized travel-times and thus have a lower probability of reaching the destination on time.*

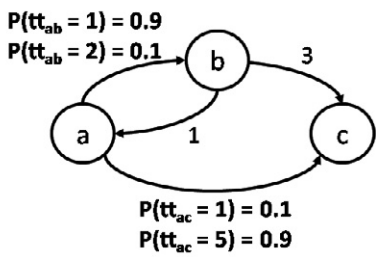
[1] In this formulation of the problem, the traveler is not allowed to wait at any of the intermediate nodes. In Section 3.2 we state the conditions under which travel-time distributions from traffic information systems satisfy the first-in-first-out (FIFO) condition, and thus waiting at a node cannot improve the on time arrival probability of the modeled traveler.

Algorithm 1 Successive approximations algorithm (Fan and Nie (2006))

Step 0. Initialization
$k = 0$
$u_i^k(t) = 0, \ \forall i \in N, \ i \neq s, \ 0 \leq t \leq T$ % $u_i^k(t)$ is the approximation of $u_i(t)$
$u_s^k(t) = 1, \ 0 \leq t \leq T$ in the k^{th} iteration of the algorithm

Step 1. Update
$k = k + 1$
$u_s^k(t) = 1, \ 0 \leq t \leq T$
$u_i^k(t) = \max_j \int_0^t p_{ij}(\omega)\, u_j^{k-1}(t-\omega)\, d\omega, \forall i \in N, \ i \neq s, \ (i,j) \in A, \ 0 \leq t \leq T$

Step 2. Convergence test
If $\forall (i,t) \in N \times [0,T]$, $\max_{i,t} |u_i^k(t) - u_i^{k-1}(t)| = 0$ stop;
Otherwise go to Step 1.



Path	Travel-time	Probability
$\{(a,b),(b,c)\}$	4	0.9
$\{(a,c)\}$	1	0.1
$\{(a,b),(b,a),(a,c)\}$	4	0.01

Figure 1: A simple network with an optimal routing policy that may contain a loop. Links (b,c) and (b,a) have deterministic travel-times of respectively 3 and 1 time units. Link (a,b) has a travel-time of 1 with probability 0.9 and a travel-time of 2 with probability 0.1. Link (a,c) has a travel-time of 5 with probability 0.9 and a travel-time of 1 with probability 0.1. The table presents on time arrival probabilities for all possible realizations of optimal paths.

As stated in Fan and Nie (2006), as the network gets arbitrarily complex, it is possible to have an infinite-horizon routing process. Bertsekas and Tsitsiklis (1996) showed that this is unlikely to happen in realistic networks, but whether a successive approximations solution to this problem converges in a finite number of steps is an open problem to the best of our knowledge.

To solve the problem raised by the unbounded convergence of Algorithm 1, Nie and Fan (2006) present a discrete time approximation of the SOTA problem. This algorithm has a computational complexity of $O(m(Td)^2)$, where m is the number of links in the graph, d is the number of discretization intervals per unit time and T is the time budget. The drawback of this method is the numerical discretization error in the representation of the probability density function. A smaller discretization interval leads to a more accurate approximation, but increases the computation time quadratically. In this article, we present both a continuous time exact solution that does not require successive approximations and a discretization scheme with a computation time to approximation error trade off that is lower than the standard discretization scheme in most practical cases.

3. Continuous time exact formulation of the SOTA problem with single iteration convergence algorithm

In this section, we present an algorithm that finds the optimal solution to the continuous time SOTA problem in a single iteration through time space domain of the problem. The complexity of the algorithm does not depend on the number of links in the optimal path.

3.1. Solution algorithm for single iteration convergence

The key observation used in this algorithm is that there exists a minimum physically realizable travel-time on each link of the network. Let β be the minimum realizable link travel-time across the entire network. β is strictly positive since speeds of vehicles have a finite uniform bound, and the network contains a finite number of links with strictly positive length. Therefore, given $\epsilon \in (0, \beta)$, $\delta = \beta - \epsilon$ is a strictly positive travel-time such that $p_{ij}(t) = 0 \ \forall\, t \leq \delta, (i, j) \in A$. Given a time budget T discretized in intervals of size δ, let $L = \lceil T/\delta \rceil$. We propose the solution Algorithm 2.

Algorithm 2 Single iteration SOTA algorithm

Step 0. Initialization.
$k = 0$
$u_i^k(t) = 0, \ \ \forall i \in N, \ \ i \neq s, \ \ t \in [0, T)$
$u_s^k(t) = 1, \ \ \forall\, t \in [0, T)$

Step 1. Update
For $k = 1, 2, \ldots, L$
  $\tau^k = k\delta$
  $u_s^k(t) = 1, \ \ \forall\, t \in [0, T)$
  $u_i^k(t) = u_i^{k-1}(t)$
    $\forall i \in N, \ \ i \neq s, \ \ t \in [0, \tau^k - \delta]$
  $u_i^k(t) = \max_j \int_0^t p_{ij}(\omega)\, u_j^{k-1}(t - \omega)\, d\omega$ % computation of the convolution product
    $\forall i \in N, \ \ i \neq s, \ \ (i, j) \in A, \ \ t \in (\tau^k - \delta, \tau^k]$

In this formulation of the SOTA problem, the functions $u_i^k(\cdot)$ are computed on $[0, T]$ by increments of size δ. The proposed algorithm relies on the fact that for $t \in (\tau^k - \delta, \tau^k]$, $u_i^k(t)$ can be computed exactly using only $u_j^{k-1}(\cdot)$, $(i, j) \in A$, on $(\tau^k - 2\,\delta, \tau^k - \delta]$, where τ^k is the budget up to which $u_i^k(\cdot)$ is computed at the k^{th} iteration of Step 1.

Proposition 1. *Algorithm 2 finds the optimal policy for the SOTA problem in a single iteration.*

Proof 1. *Proof by induction over the sub-steps $\tau = \delta$ to $L\delta$.*

Base case: When $k = 1$ the convolution product is computed on the interval $(0, \delta]$. From the definition of δ, we know that there does not exist a realizable travel-time that is less than or equal to δ. Therefore, $p_{ij}(\omega) = 0$ on the interval $(0, \delta]$ of the convolution product, and $u_i^1(t) = u_i^0(t)\ \forall i \in N$. This is indeed the correct solution $\forall\, t$ such that $0 \leq t \leq \delta$, since no feasible path to the destination exists in this time interval.

Induction step: Assume that the algorithm generates an optimal policy for $k < L$. We show that the algorithm also provides an optimal policy at step $k + 1$. When $t = (k + 1)\delta$, the convolution product is computed on the interval $(k\delta, (k+1)\delta]$. Since $p_{ij}(\omega) = 0, \forall\, \omega$ such that $\omega \leq \delta$, to find the optimal policy for $u_i^{k+1}(t), \forall\, t$ such that $k\delta < t \leq (k + 1)\delta$, we only need to know the optimal policy for $u_j^k(t), \forall\, t$ such that $0 \leq t \leq k\delta$, $(i, j) \in A$. By the induction hypothesis we know that $u_i^k(t)$ gives the optimal policy $\forall\, t$ such that $0 \leq t \leq k\delta$ for all nodes and it is known. This implies that the optimal policy is computed for the range $0 \leq t \leq (k + 1)\delta$ at the end of the $(k + 1)^{th}$ step. ■

The most computationally intensive step of this algorithm is the computation of the convolution product, which is represented algebraically in the above algorithm. If the link travel-time distributions $p_{ij}(\cdot)$, $(i, j) \in A$, and the cumulative optimal travel-time distributions $u_i(\cdot)$, $i \in N$, belong to a parametric distribution family which is closed under convolution (e.g. Gaussian, Erlang, see Block and Savits (1976); Assaf and Levikson (1982)), the convolution product can be computed analytically. However, since $u_i(\cdot)$ is the point wise maximum of the convolution products of the link travel-time distributions $p_{ij}(\cdot)$ and the cumulative distributions $u_j(\cdot)$, $(i, j) \in A$, it does not have an analytical expression in general.

Numerical approximations of the distributions involved in the convolution product have been proposed in the literature. Fan et al. (2005) argue that since $u_i(\cdot)$ is a continuous monotone increasing function, it can be approximated by a low degree polynomial. When the approximating polynomial is of degree $2n$, the convolution integral can be solved exactly with n evaluation points using the Gaussian quadrature method (see Bellman and Kalaba (1966)).

The applicability of these methods is highly dependent on the shape of the travel-time distributions, which can be very complex, depending on the traffic conditions and the topology of the network. Therefore, we propose solving the convolution product via a time discretization of the distributions involved, which results in a computational complexity that is independent of the shape of the optimal cumulative travel-time distributions $u_i(\cdot)$. An incremental update scheme that exploits the structure of the proposed SOTA algorithm is used to efficiently compute the discrete convolution product. This solution is shown to be computationally less expensive than than the existing convolution methods for the SOTA problem (e.g Nie and Fan (2006)). A detailed explanation is given in Section 4.

3.2. Extended algorithm for time-varying link travel-times

The solution algorithm proposed in the previous section makes the assumption that link travel-time distributions are static. However, in real transportation networks, it is clear that link travel-time distributions are time-varying. In this section, we present an extension to Algorithm 2 that accounts for time-varying distributions. A common approach when solving shortest path problems on graphs with time-dependent distributions is to consider the corresponding time expanded graph with static weights. See Dean (2004) for a discussion on the various flavors of this problem. If the *first-in-first-out* (FIFO) condition holds, the problem can be solved without time-expanding the graph using a trivially modified version of Dijkstra's algorithm and indexing the link weights by time (see Dreyfus (1969)). We propose a similar algorithm based on the fact that waiting at a node is never optimal when the FIFO conditions holds. First we show that most commonly used travel-time estimates satisfy the the FIFO condition and then show that waiting at a node is never optimal in such a model.

Definition 2. *In a deterministic setting, let $\alpha^t_{\mathcal{P}}$ denote the travel-time on path[2] $\mathcal{P}$ when departing at time t. The graph satisfies the FIFO condition if and only if:*

$$\alpha^{t_1}_{\mathcal{P}} \leq \alpha^{t_2}_{\mathcal{P}} + (t_2 - t_1) \;\; \forall \;\; path \;\; \mathcal{P} \;\; and \;\; \forall \, t_1, t_2 \;\; such \; that \;\; 0 \leq t_1 \leq t_2 \tag{2}$$

This definition states that on a given path $\mathcal{P}$, the travel-time $\alpha^{t_1}_{\mathcal{P}}$ when leaving at t_1 is lower than the travel-time $\alpha^{t_2}_{\mathcal{P}} + (t_2 - t_1)$ obtained by waiting at the departure node for $t_2 - t_1$ and departing at t_2.

In the time-varying setting, the link travel-time estimates given by a travel-time model can change as a vehicle moves through a link. We assume an elastic vehicle travel-time model, where the vehicle link travel-time is calculated based on all the link travel-time estimates the vehicle might encounter as it moves through a link. This is in contrast to a frozen vehicle travel-time model, where the travel-time is calculated simply based on the link travel-time estimate when the vehicle enters the link[3].

Proposition 2. *Under an elastic vehicle travel-time model, a deterministic discrete-time traffic estimate such that link travel-time is single-valued on each time discretization yields a deterministic FIFO path.*

Proof 2. *Consider two vehicles traveling along the same path $\mathcal{P}$ from node i to node k departing at times t_1 and t_2 respectively for $0 \leq t_1 \leq t_2$. Their respective travel-times are denoted $\alpha^{t_1}_{\mathcal{P}}$ and $\alpha^{t_2}_{\mathcal{P}}$. For the vehicle departing node i at time t_2 to arrive at node k before the vehicle departing at time t_1, it must overtake the vehicle that departed first. Overtaking can only occur when both vehicles are in the same space-time cell. However, since we assume that the model gives a single-valued speed in each space-time cell, both vehicles will travel at the same speed when in the same cell and no overtaking can occur. Therefore, a single-valued speed in each space-time cell implies that the vehicle that departed first will always arrive first.* ■

[2]The FIFO condition is typically defined on a link. Here, we use a path-based definition to make the subsequent explanations and proofs more intuitive. This leads to equivalent results under the assumption that the network topology is static.

[3]Please see Orda and Rom (1990) for further discussion on the elastic and frozen travel-time models.

This guarantees that the shortest path problem on transportation networks, with time-varying link travel-times generated by a traffic information system, can be solved with the same complexity as in the static case by time-indexing the link weights (Dean (2004)). In the case of transportation networks with stochastic link travel-times, for the SOTA problem (equation (1)), a similar stochastic FIFO condition is needed to guarantee correctness of the algorithm with time-indexed link travel-times.

Definition 3. *Let $u^t_{\mathcal{P}}(\cdot)$ denote the cumulative travel-time distribution on path $\mathcal{P}$ when departing at time t. The graph satisfies the stochastic FIFO condition if and only if:*

$$u^{t_1}_{\mathcal{P}}(T) \geq u^{t_2}_{\mathcal{P}}(T-(t_2-t_1)) \;\; \forall \;\; path \;\; \mathcal{P} \;\; and \;\; \forall\, T, t_1, t_2 \;\; such\; that \;\; 0 \leq t_1 \leq t_2, \;\; t_2 - t_1 \leq T \tag{3}$$

This definition states that at any given time and on any given path on the network, departing as soon as possible yields a greater probability of arriving on time than delaying the departure. The stochastic FIFO property is obtained if the conditions defined by Proposition 3 are satisfied.

Proposition 3. *A stochastic discrete-time traffic estimate such that link travel-time distributions are fixed for each time discretization yields a stochastic FIFO path.*

Proof 3. *Proof by induction over the length of the path $(v_n, \ldots, v_1)$.*

Base case ($n = 2$): From definition 3, a stochastic FIFO network satisfies the following condition:

$$u^{t_1}_{v_2v_1}(T) \geq u^{t_2}_{v_2v_1}(T-(t_2-t_1)) \;\; \forall \;\; 0 \leq t_1 \leq t_2$$

where $u^t_{ij}(T)$ is the probability of arriving at node j in time T when departing from node i at time t. We want to show that a delayed departure cannot improve the probability of on time arrival when traveling from node v_2 to node v_1 on link (v_2, v_1). Without loss of generality, let vehicle w_1 depart from node v_2 at time t_1 and vehicle w_2 depart from node v_2 at time t_2, where w_1 is in cell $c_1 = (v_2, v_1) \times [t_{c_1}, t_{c_2}]$ and w_2 is in cell $c_2 = (v_2, v_1) \times [t_{c_2}, t_{c_3}]$. Also, let $X_w(t_a, t_b)$ be the distribution of the distance traveled by vehicle w in the interval (t_a, t_b). We have the following:

$$\begin{aligned} X_{w_2}(t_1, t_2) &= 0 \\ X_{w_1}(t_1, t_2) &\geq 0 \end{aligned}$$

Let x be the length of link (v_2, v_1). We want to show that:

$$\begin{aligned} & u^{t_1}_{v_2v_1}(T) \geq u^{t_2}_{v_2v_1}(T-(t_2-t_1)) \\ \iff \quad & P(X_{w_1}(t_1, t_1+T) \geq x) \geq P(X_{w_2}(t_2, t_1+T) \geq x) \\ \iff \quad & P(X_{w_1}(t_1, t_{c_2}) + X_{w_1}(t_{c_2}, t_2) + X_{w_1}(t_2, t_1+T) \geq x) \geq P(X_{w_2}(t_2, t_1+T)) \geq x) \end{aligned}$$

This clearly holds because:

$$X_{w_1}(t_2, t_1+T) = X_{w_2}(t_2, t_1+T)$$

since both vehicles are in the same space-time cells from time t_2.

Induction step ($n = k$): We assume that a single-valued travel-time distribution for each space-time cell implies a stochastic FIFO path $(v_k, \ldots, v_1)$ of k nodes and show that it holds for $k+1$ nodes. The explanation in the base case shows that departing from node v_{k+1} earlier gives a higher probability of reaching node v_k within a given time budget. The induction hypothesis implies that arriving at node v_k earlier increases the probability of reaching the destination v_1 within a given time budget. Therefore, leaving the node v_{k+1} earlier increases the probability of reaching the destination within a given time budget. ■

Under the stochastic FIFO condition, we now show that an optimal policy for the SOTA problem does not prescribe waiting at a node. Therefore, the SOTA problem on transportation networks with time-varying link travel-time distributions can be solved by time-indexing the link travel-time distributions.

Proposition 4. *In a stochastic FIFO network, according to the optimal policy for the SOTA problem, waiting at a non-terminal node is not optimal.*

Proof 4. *Proposition 3 shows that waiting at a node cannot improve the probability of arriving within a certain budget using the same path. We now show that waiting at a node cannot improve the probability of arriving within a given budget T when using the optimal path for each departure time. Assume that path $\mathcal{P}_1$ is the optimal path when departing at time t_1 and that path $\mathcal{P}_2$ is the optimal path when departing at time t_2. From proposition 3 we know that $u^{t_1}_{\mathcal{P}_2}(T) \geq u^{t_2}_{\mathcal{P}_2}(T-(t_2-t_1))$. Furthermore, since $\mathcal{P}_1$ is the optimal path when leaving at time t_1 with a budget of T, we have $u^{t_1}_{\mathcal{P}_1}(T) \geq u^{t_1}_{\mathcal{P}_2}(T)$. Therefore, $u^{t_1}_{\mathcal{P}_1}(T) \geq u^{t_2}_{\mathcal{P}_2}(T-(t_2-t_1))$ and waiting cannot improve the probability of on time arrival.* ■

When the assumptions from Proposition 3 are satisfied, according to Proposition 4, waiting at a node is not part of the optimal policy. Therefore, the optimal policy in the time-varying case can be defined as follows:

$$\begin{aligned} u^{\tau}_i(t) &= \max_j \int_0^t p^{\tau}_{ij}(\omega)\, u^{\tau+\omega}_j(t-\omega)\, d\omega \\ &\quad \forall i \in N,\ \ i \neq s,\ \ (i,j) \in A,\ \ 0 \leq t \leq T,\ \ 0 \leq \tau \\ u^{\tau}_s(t) &= 1 \quad \forall\ 0 \leq t \leq T, 0 \leq \tau. \end{aligned} \tag{4}$$

where $u^{\tau}_i(t)$ is the maximum probability of arriving at destination s within time budget t when leaving node i at time τ. Waiting is not allowed in the optimal policy as enforced by the fact that the departure time from node i (superscript of $u^{\tau}_i(\cdot)$) is the same as the time at which the link (i, j) is traversed (superscript of $p^{\tau}_{ij}(\cdot)$). This policy is optimal according to Proposition 4.

The proposed algorithm uses the same network as the static SOTA problem and simply replaces the travel-time distribution query with a time-indexed version, as defined in Equation 4. i.e. it modifies the convolution step to query the appropriate link travel-time distribution based on the current time offset τ. This algorithm has the same computational complexity as the static algorithm because the structure of the graph remains the same and no additional queries are performed. The required memory is larger in the time-varying case than in the static case because there are multiple travel-time distributions for each link.

3.3. Generalized algorithm for correlated link travel-times

In this section, we extend the capabilities of the previous algorithm, by relaxing another assumption that was made when presenting the algorithm in Section 2. The formulation presented so far assumes that the link travel-times on the network are uncorrelated. However, in reality the travel-times of neighboring links are correlated. Assuming that link travel-times satisfy the Markov condition, they only depend on their upstream and downstream neighbors. Therefore, the travel-time on each link is a joint distribution over the link and its neighbors. If we do not have any information regarding the travel-times on these other links, the correct approach is to marginalize them out and use the marginal distribution of the link we are considering. However, in the SOTA formulation each decision could be preceded by conditioning on the travel-time of an upstream link. Assuming independence in this case results in an inaccurate expected travel-time and an overestimation of the variance. Therefore, to minimize such errors one must incorporate this observation and use the conditional probability distribution of the link travel-time. We present a simple extension to our formulation that considers the correlation between a link and the upstream neighbors via which the link is reached. The proposed problem formulation is as follows:

$$
\begin{aligned}
u_i(t,k,y) &= \max_j \int_0^t p_{ij}(tt_{ij} = \omega | tt_{ki} = y)\, u_j(t-\omega, i, \omega)\, d\omega \\
&\quad \forall i \in N,\ \ i \neq s,\ \ (i,j) \in A,\ \ (k,i) \in A, \\
&\quad 0 \leq y \leq (T-t),\ \ 0 \leq t \leq T \\
u_s(t,k,y) &= 1 \quad \forall\ 0 \leq t \leq T,\ \ (k,s) \in A,\ \ 0 \leq y \leq T-t
\end{aligned} \tag{5}
$$

where $u_i(t,k,y)$ is the cumulative distribution function (CDF) when t is the remaining time budget, k is the node from which the vehicle is arriving and y is the realized travel-time on this upstream link, and $p_{ij}(tt_{ij} = \omega | tt_{ki} = y)$ is the probability that the travel-time on link (i,j) is ω conditioned on the travel-time on link (k,i) being y. The joint probability density function of a link and its upstream neighbors is assumed to be known. In this case, each CDF $u_i(\cdot,\cdot,\cdot)$ and travel-time distribution $p_{ij}(\cdot)$ is now conditioned on the upstream travel-time, but the structure of the problem remains unchanged. We are simply propagating more information at each step. Therefore, the correctness analysis of our algorithm remains unchanged and the same proof holds. However, the time complexity of the algorithm increases since a new dimension is being added to the problem. Equation (5) needs to be solved for each incoming node k and the travel-time on link (k,i), which is in the range $0 \leq y \leq (T-t)$. This increases the runtime of the original formulation (Equation (1)) by a factor of ΦT, where Φ is the maximum in-degree of the network.

In most practical cases this is likely to make the algorithm intractable for real-time applications, as the complexity is now cubic in T. Therefore, we propose using a discrete approximation of the conditional distribution function. In the simplest case, the conditioning can be done based on whether the upstream link was in congestion or free flow, which will only increase the complexity by a factor of 2Φ. If upstream travel-time is discretized in to d ranges, the increase in complexity will be a factor of $d\Phi$. The most appropriate value to use for d depends on the quality of the conditional travel-time distributions available and the computing resources that can be utilized.

4. Discrete formulation of the SOTA algorithm with a Fast Fourier Transform solution

For practical networks, a numerical approximation of the convolution integral is necessary, with a proper discretization. In the discrete setting, the proposed algorithm can be formulated as shown in Algorithm 3 below.

Algorithm 3 Discrete SOTA algorithm

Step 0. Initialization.
$k = 0$
$u_i^k(x) = 0,\ \ \forall i \in N,\ \ i \neq s,\ \ x \in \mathbb{N},\ \ 0 \leq x \leq \frac{T}{\Delta t}$
$u_s^k(x) = 1,\ \ x \in \mathbb{N},\ \ 0 \leq x \leq \frac{T}{\Delta t}$

Step 1. Update
For $k = 1, 2, \ldots, L$
  $\tau^k = k\delta$
  $u_s^k(x) = 1,\ \ x \in \mathbb{N},\ \ 0 \leq x \leq \frac{T}{\Delta t}$
  $u_i^k(x) = u_i^{k-1}(x)$
    $\forall i \in N,\ \ i \neq s,\ \ (i,j) \in A,\ \ x \in \mathbb{N},\ \ 0 \leq x \leq \frac{(\tau^k - \delta)}{\Delta t}$
  $u_i^k(x) = \max_j \sum_{h=0}^{x} p_{ij}(h)\, u_j^{k-1}(x-h)$
    $\forall i \in N,\ \ i \neq s,\ \ (i,j) \in A,\ \ x \in \mathbb{N},\ \ \frac{(\tau^k - \delta)}{\Delta t} + 1 < x \leq \frac{\tau^k}{\Delta t}$ % Δt is selected such that $\delta > \Delta t$

where Δt is the length of a discretization interval and T is the time budget. The functions $u_i(\cdot)$ and $p_{ij}(\cdot)$ are vectors of length $L = \lceil \frac{T}{\Delta t} \rceil$. For notational simplicity, we assume that T is a multiple of Δt. In general, the link travel-time distributions are available as either discrete or continuous time distributions. If the link travel-time distribution is

discrete and the length of the discretization interval d is not equal to Δt, the probability mass needs to be redistributed to intervals of Δt. If the distribution is continuous, the probability mass function $p_{ij}(\cdot)$ is computed as follows:

$$p_{ij}(h + \Delta t) = \int_{h}^{h+\Delta t} p_{ij}(\omega)\ d\omega, \quad \forall h = 0, \Delta t, \ldots, (L-1)\Delta t \tag{6}$$

4.1. Complexity analysis

Obtaining the appropriately discretized probability mass functions can be done in time $O(\frac{mT}{\Delta t})$, since there are m links and each link travel-time distribution function is of length $\frac{T}{\Delta t}$. This can also be computed in advance and reused during each call to the algorithm[4]. In step 0, initializing k vectors (one for each node i) of length $\frac{T}{\Delta t}$ takes $O(\frac{kT}{\Delta t})$ time. In step 1, notice that for each link (i, j) the algorithm progressively computes a sum of increasing length from $x = 1$ to $x = \frac{L\delta}{d} = \frac{T}{\Delta t}$. Therefore, the time complexity of the summation for each link is $O((\frac{T}{\Delta t})^2)$. The assignment $u_i^k(x) = u_i^{k-1}(x)$ can be done in constant time by manipulating pointers instead of a memory copy or by simply having one array for all $u_i(\cdot)$ that keeps getting updated at each iteration of the loop. Since there are m links, the total time complexity of step 1 is $O(m(\frac{T}{\Delta t})^2)$. This dominates the complexity of step 0 and therefore is the total time complexity of the entire algorithm. Recall that this is identical to the time complexity of the discrete approximation algorithm proposed by Nie and Fan (2006).

Algorithm 3 can perform more efficiently by computing the convolution products via the *Fast Fourier Transform* (FFT). The FFT computes the convolution of two vectors of length n in $O(n \log(n))$ time, see Cormen et al. (2001). Notice however that the proposed algorithm does not compute the entire convolution at once. The computation is required to be done in blocks of length δ to preserve optimality. Therefore, L convolution products of increasing length $\delta, 2\delta, \ldots, L\delta$ have to be computed. One inefficiency of this approach is that successive convolutions recompute the results that have already been obtained. However, using the FFT can still be shown to perform better than a brute force convolution product with quadratic complexity. For each link, the time complexity of the sequence of FFTs is $O(\sum_{k=1}^{L} \frac{\delta k}{\Delta t} \log(\frac{\delta k}{\Delta t}))$, where $L = \lceil \frac{T}{\delta} \rceil$. Since there are m links, the total time complexity is:

$$O\left(m \sum_{k=1}^{\lceil \frac{T}{\delta} \rceil} \frac{\delta k}{\Delta t} \log\left(\frac{\delta k}{\Delta t} \right) \right) \tag{7}$$

As $T \to \infty$, the complexity of the FFT based approach $O((\frac{T}{\Delta t}^2 \log(\frac{T}{\Delta t}))$ is asymptotically larger than the run-time of the brute force approach $\sum_{k=1}^{\frac{T}{\Delta t}} k = O((\frac{T}{\Delta t})^2)$. However, the running time of the FFT approach is significantly smaller than the brute force approach in the time range of interest for most practical applications, as shown in proposition 5.

Proposition 5. *The travel budget $t = \Delta t \cdot 2^{\frac{1}{4}(3+\frac{\delta}{\Delta t})} - \delta$ is a lower bound for the largest budget at which the FFT based approach has a faster run-time than the brute force approach. See Appendix A for proof.*

This lower bound t is typically a large value in road networks where individual links have large travel-times. Table 1 shows the value of t for some sample values of δ and Δt. The exact value of t can be obtained by computing summation (7) numerically.

4.2. Acceleration of Algorithm 3 with localization

As shown in Section 4.1, the runtime of the FFT based solution is a function of δ and decreases as the value of δ increases. The value of δ that is used in the algorithm is bounded by the minimum realizable travel-time across the entire network. However, in general, road networks are heterogeneous and contain a large range of minimum realizable travel-times. This section presents an optimization that can significantly improve the runtime of the proposed algorithm by exploiting the disparity of these local δ values.

[4] In the case of time-varying link travel-times, this needs to be recomputed for each time step at which the travel-times differ.

	$\Delta t = 0.1$	$\Delta t = 0.2$	$\Delta t = 0.5$	$\Delta t = 1$
$\delta = 90$	$1.51 \cdot 10^{65}$	$4.11 \cdot 10^{31}$	$4.93 \cdot 10^{11}$	$1.6 \cdot 10^{5}$
$\delta = 60$	$4.00 \cdot 10^{42}$	$2.11 \cdot 10^{20}$	$1.50 \cdot 10^{7}$	$9.17 \cdot 10^{2}$
$\delta = 30$	$1.06 \cdot 10^{20}$	$1.08 \cdot 10^{9}$	$4.59 \cdot 10^{2}$	4.57
$\delta = 15$	$5.44 \cdot 10^{8}$	$2.47 \cdot 10^{3}$	2.41	$1.27 \cdot 10^{-1}$

Table 1: Lower bound $t = \Delta t \cdot 2^{\frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)} - \delta$ (minutes) for which the FFT approach is faster than the brute force approach for the computation of the convolution product in the main algorithm. Values of δ and Δt are given in seconds.

Proposition 6. *Let β_{ij} be the minimum realizable travel-time on link (i, j) with $\delta_{ij} = \beta_{ij} - \epsilon$ $(0 < \epsilon < \beta_{ij})$ and τ_i be the budget up to which the cumulative distribution function $u_i(\cdot)$ has been computed for node i. For correctness, the invariant*

$$\tau_i \leq \min_j(\tau_j + \delta_{ij}) \quad \forall (i, j) \in A \tag{8}$$

must be satisfied throughout the execution of the algorithm.

Proof 6. *Assume that this invariant can be violated. Then, it is possible to compute the cumulative distribution function $u_i(\cdot)$ at some node i such that $\tau_i > \min_j(\tau_j + \delta_{ij})$, which in turn means that $\tau_i - \tau_j > \delta_{ij}$ for at least one node j. This implies that $\exists\, t'$ such that $u_i(t')$ was computed using the product of a downstream cumulative distribution function $u_j(t' - \omega)$ and $p_{ij}(\omega)$, where $u_j(t' - \omega)$ is unknown because $\tau_j < t' - \delta_{ij}$ and $p_{ij}(\omega) > 0$ because $\omega > \delta_{ij}$. This value of the cumulative distribution function $u_i(t')$ is undefined and the SOTA algorithm fails. Therefore, for correctness the invariant should not be violated.* ■

When computing the cumulative density function $u_i(\cdot)$ using local δ_{ij} values, the growth of τ_i is different across the nodes i, unlike in our original algorithm (Algorithm 3) where the τ_i grow at the constant uniform rate δ. Furthermore, when $u_i(\cdot)$ is updated asynchronously using the invariant $\tau_i \leq \min_j(\tau_j + \delta_{ij})$, $(i, j) \in A$, the order in which the nodes are updated impacts the runtime of the algorithm, as illustrated in Example 2.

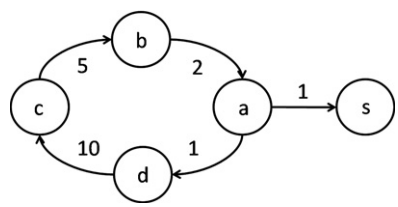


Figure 2: Example of a simple loop. The δ value for each link is given along the link.

Example 2. *To illustrate how the order in which the nodes are updated impacts the runtime of Algorithm 3, consider the network in Figure 2. The value of τ_i and the computation time depends on the order in which the nodes are considered. In the worst case, as a lower bound, we assume that $u_i(\cdot)$ is updated based on the values of its constraint nodes in the previous iteration. Table 2 shows the sequence of updates for four iterations when using the constraints τ_i from the previous iteration. Notice that the update pattern is cyclic every four iterations. The highest speedup is achieved when the nodes in the loop are considered in topological order. Table 3 shows the sequence of updates when the nodes are considered in the topological order (a, b, c, d). As seen in Table 3, the τ_i value for each node i can be incremented by the length of the shortest loop node i belongs to when the nodes are updated in this order. The topological order can be determined easily in this simple example, but such an ordering is unlikely to exist in realistic transportation networks. Table 4 shows the sequence of updates when the nodes are considered in the order (d, c, b, a). It can be clearly seen that this ordering is much more inefficient than the ordering (a, b, c, d). Furthermore, without the local δ optimization, the algorithm can only update u_i by one step at each iteration, since the minimum*

δ_i value is 1 in this example. This simple example shows how local δ optimization can provide large improvements in the runtime.

Iter.	a	b	c	d
1	1	2	5	10
2	11	3	7	15
3	16	13	8	17
4	18	18	18	18

Table 2: τ_i values when computing u_i constrained on previous iteration.

Iter.	a	b	c	d
1	1	3	8	18
2	19	21	26	36
3	37	39	44	54
4	55	57	62	72

Table 3: τ_i values when computing u_i in the order (a, b, c, d).

Iter.	d	c	b	a
1	10	5	2	11
2	15	7	13	16
3	17	18	18	18
4	28	23	20	29

Table 4: τ_i values when computing u_i in the order (d, c, b, a).

Given that the runtime of the SOTA algorithm depends on the update order, we would like to find an optimal ordering that minimizes the runtime of the algorithm. The first step in finding such an optimal ordering is to formalize the runtime of the FFT SOTA algorithm.

Definition 4. *The computation time of the cumulative density function $u_i(\cdot)$ can be minimized by finding the ordering that solves the following optimization problem.*

$$
\begin{aligned}
&\text{minimize}_{(\tau_i^{k_i}, K_i)} && \sum_{(i,j)\in A}\sum_{k_i=1}^{K_i} \frac{\tau_i^{k_i}}{\Delta t}\log\frac{\tau_i^{k_i}}{\Delta t} && && (9)\\
&\text{subject to} && \tau_i^{k_i} \le \tau_j^{k_j} + \delta_{ij} && \forall \tau_i^{k_i}, \tau_j^{k_j} \ \ s.t.\ \ (i,j)\in A,\\
& && && C(i,k_i) < C(j, k_j+1)\\
& && \tau_r^{K_r} \ge T &&\\
& && \tau_s^{1} \ge T &&\\
& && \tau_i^{1} \ge \Delta t && \forall i \in N, i \neq s\\
& && \tau_i^{k+1} > \tau_i^{k} && \forall i \in N
\end{aligned}
$$

where $\tau_i^{k_i}$ is the budget up to which $u_i(\cdot)$ has been computed in the k_i^{th} iteration of computing $u_i(\cdot)$, $C(\cdot,\cdot)$ is an index on the order in which nodes are updated such that $C(i, k_i)$ denotes when node i is updated for the k_i^{th} time and K_i is the total number of iterations required for node i.

The optimal order in which $u_i(\cdot)$ is computed might result in updating some set of nodes multiple times before updating another set of nodes.

Proposition 7. *The ordering that gives the optimal solution to the optimization problem (9) can be obtained using Algorithm 4 in $O(\frac{mT}{\Delta t}\log(n))$ time, where n and m are respectively the number of nodes and links in the network, Δt is the time discretization interval and T is the time budget. See Appendix B for proof.*

The optimal order of updates (node and value) that computes the cumulative distribution function $u_r(T)$ of the origin r most efficiently is stored in the stack χ at the termination of the algorithm. Algorithm 4 works by taking the source node r and the time budget to which it needs to be updated T, and then recursively updating the set of constraints that need to be satisfied before $u_r(T)$ can be computed. At the first iteration, the source and its terminal value in the algorithm (the budget) are added to the stack, and the constraints that are required for updating the source to that value are stored in the heap ψ. At any given iteration, the largest value in the heap is extracted and added to the stack, since it is the most constrained node in the current working set. We leave further discussion of how the algorithm works to the proof of correctness in Appendix B.

Algorithm 4 Optimal order for updating $u(\cdot)$

Step 0. Initialization.
$\tau(i) = 0, \quad \forall i \in N, \quad i \neq r, \quad i \neq s$ % where r is the origin and s is the destination
$\tau(s) = \infty, \quad \tau(r) = T$
$\psi := \{(r, \tau(r))\}$ % ψ is a priority queue data structure
$\chi := \{(r, \tau(r))\}$ % χ is a stack data structure

Step 1. Update
(v, θ) =ExtractMax(ψ)
$\tau(v) := \theta$
Push($\chi, (v, \tau(v))$)
π := Children(v)
For $k := 1$ to size(π)
  If (($\pi[k] \neq s$) and ($\tau(v) - \delta_{v\pi[k]}) > 0$)
    τ := max(Extract($\psi, \pi[k]$), $\tau(v) - \delta_{v\pi[k]}$)
    Insert($\psi, (\pi[k], \tau)$)

Step 2. Termination
If $\psi := \emptyset$ stop;
Otherwise go to Step 1.

4.3. Search space pruning by elimination of infeasible paths

The proposed algorithm requires computing the cumulative distribution function $u_i(t)$ for every node in the graph. This can be prohibitively expensive even in reasonably sized road networks. Therefore, we need to constrain the search space of our algorithm. In this section, the proposed algorithm is extended by adding a pruning algorithm that eliminates infeasible paths by removing unnecessary nodes during a preprocessing step. Consider an instantiation of the SOTA problem with an origin node r, destination node s and a travel-time budget of T. Since every link (i, j) in the network has a minimum realizable travel-time β_{ij} (as defined in Section 3.1), it follows that every path in the network must have a minimum realizable travel-time as well. For any path $\mathcal{P}_{ik}$, let the minimum realizable path travel-time be α_{ik}. The value of α_{ik} can be found by running a standard deterministic shortest path algorithm such as Dijkstra's algorithm on the network with the link weights being the minimum realizable link travel-time corresponding to each link.

Proposition 8. *Consider some arbitrary node i in the network. Let α_{ri} and α_{is} be respectively the minimum realizable travel-times from the origin to node i and from node i to the destination.*

1. *If $\alpha_{ri} + \alpha_{is} > T$, we can safely remove this node from the network and ignore it when solving the SOTA problem.*
2. *The cumulative distribution function $u_i(\cdot)$ only needs to be computed for the time interval $\alpha_{is} \leq t \leq T - \alpha_{ri}$.*

Proof 8.

1. *If $\alpha_{ri} + \alpha_{is} > T$, the minimum realizable travel-time from the origin to the destination through node i is greater than the travel-time budget. Therefore, no feasible path exists through node i.*
2. *The minimum realizable travel-time from the origin to node i is α_{ri}. Therefore, no path in the dynamic programming recursion will query $u_i(t)$ for $t > T - \alpha_{ri}$. The minimum realizable travel-time from node i to the destination is α_{is}. Therefore, $u_i(t)$ is zero for $t < \alpha_{is}$.* ■

By performing an all destinations shortest path computation from the source and an all sources shortest path computation from the destination, we can significantly prune both the size of the network required when solving the SOTA problem and the time interval for which the cumulative distribution function $u_i(\cdot)$ needs to be computed for each node. This pruning algorithm is inspired by the Reach heuristic (Goldberg et al. (2007)) for deterministic shortest

paths. For a graph with n nodes and m links, the time complexity of Dijkstra's algorithm is $O(m + n \log n)$ (Cormen et al. (2001)), which is dominated by the complexity of the SOTA algorithm. Thus, the cost of pruning is negligible compared to the complexity of the SOTA algorithm. Furthermore, state of the art shortest path algorithms can run in near constant time (see Abraham et al. (2010); Geisberger et al. (2008)) when the minimum realizable travel-time does not vary with time. It should be noted that this is a very conservative pruning algorithm and that further runtime reductions can be achieved using more aggressive pruning methods, which we are currently exploring.

5. Implementation of the algorithm in the *Mobile Millennium* system

Experimental results are provided by implementing the proposed algorithms in the *Mobile Millennium* traffic information system. *Mobile Millennium* is a traffic information system that integrates traffic measurements from a variety of sensors (loop detectors, radars, probe vehicles, tolltag readers) to produce real-time traffic estimates for the San Francisco Bay Area. The system collects on the order of 2 million reports of vehicle counts and time occupancy from loop detectors, count and point speeds from over 100 radars, and between $2 - 10$ million probe vehicle speeds from traffic commuters daily for the San Francisco Bay Area. These data feeds are filtered and fused using multiple techniques (viability tools (Aubin (2001)), statistical methods, etc.), and integrated as real-time observations by highway and arterial traffic estimation modules. The *Mobile Millennium* system provides real-time estimation of traffic conditions (travel-time, speed, density) on most of the non residential streets and roads in the Bay Area, on as ongoing basis. Traffic estimates are broadcast via a web interface and a cellphone application, enabling commuters to select the optimal commute route in order to avoid congestion, caused by both recurrent conditions and unexpected events such as accidents. In this section, we propose to test the performance of the routing algorithm introduced in this article on two specific traffic estimates from the system:

- Sample-based representation of the velocity map on the Bay Area highway network (Figure 3): this output is produced by the *v-CTM EnKF* algorithm from Work et al. (2010) which fuses loop detectors counts, radars speed and spatially sampled probe speeds into a partial differential equation flow model coupled with an ensemble Kalman filter estimation algorithm. This estimate is updated every 30 seconds and has a space resolution of approximately 400 meters. Link travel-time distributions representing model uncertainty on the travel-time at the mean speed can be directly computed from this output and used by our routing algorithm.

- First two moments representation of link travel-times on the San Francisco arterial network (Figure 4): this output is produced by the machine learning algorithm from Herring et al. (2010) using a mixture of real-time and historical probe generated travel-times. The link travel-time distributions represent individual commuters travel-time and can be directly used by our routing algorithm.

In the following sections, we illustrate the practical applicability of the algorithm introduced in this article in real-time traffic information systems, by implementing the algorithm in the *Mobile Millennium* system. We show that the novel optimization techniques developed in this work improve the tractability of the problem, compared to existing solutions, and make the SOTA problem feasible to solve in an operational setting.

5.1. Numerical results

The proposed algorithms are tested on the arterial and highway road networks described in the previous section. The highway network from Figure 3 is a relatively sparse network with short loops at ramps and intersections, and large loops covering the entire network. This network contains 3639 nodes and 4164 links. The San Francisco arterial network of Figure 4 is a dense network with a large number of loops with varying lengths, with 1069 nodes and 2644 links.

The algorithm is coded in Java and executed on a Windows 7 PC with a 2.67Ghz Dual Core Intel Itanium processor and 4GB of RAM. We use the open source Java libraries JTransforms (Wendykier (2009)) and SSJ (L'Ecuyer (2008)) for FFT computations and manipulating probability distributions. Time-varying link travel-time distributions are obtained a-posteriori from the traffic estimation models described above. Both travel-time models assume that the link travel-times are independent with respect to each other.

We consider the following performance metrics:

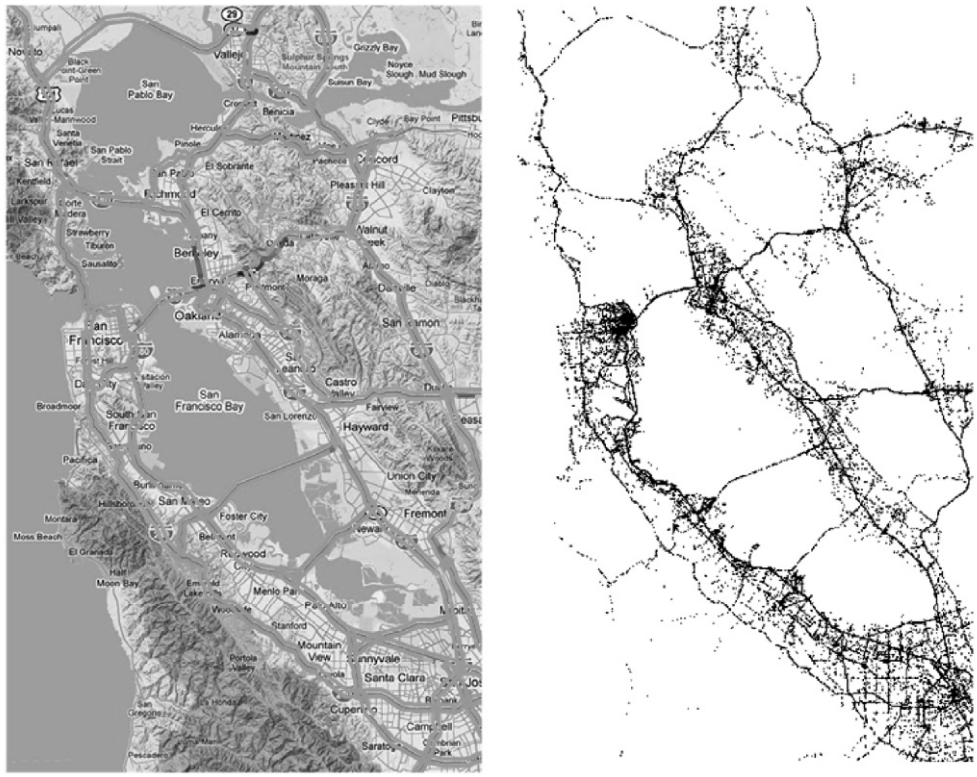


Figure 3: Best viewed in color. **Velocity estimates on the Bay Area highway network. Right:** Cumulated probe speed measurements collected on July 29th by the *Mobile Millennium* system. **Left:** Velocity estimates on July 29th at 9 pm; most of the network is in free flow, active bottlenecks correspond to red spots.

- *Runtime*: computation time for the different variants of our algorithm compared to similar existing routing algorithms and specific runtime improvement provided by the speed up techniques introduced in this work.
- *On-time arrival guarantee*: sampling the *Mobile Millennium* traffic estimates enable the generation of realistic user travel-time realizations. The probability of arriving on time for both the SOTA policy and least expected travel-time path are compared. The performance of the SOTA algorithm under different traffic conditions and route types is also analyzed.
- *En route re-routing*: performance of the algorithm on a real test case on which the ability of the routing module to provide adaptive route choices depending on traffic conditions is presented.

5.1.1. Runtime performance

The runtime of the algorithm is a critical factor in its usability for real-time routing applications. The lack of routing choices that incorporate reliability guarantees, in any of the commonly used routing applications, is in part due to the intractability of executing them efficiently. In this section, we show that the algorithms proposed in this article have the potential to bridge this gap.

As shown in the complexity analysis from Section 4.1, the algorithm introduced in this article is linear in the size of the network and pseudo-polynomial in the ratio $T/\Delta t$, where T is the time budget of the user, and Δt is the discretization interval of time. It was argued that the proposed algorithm performs better than the existing solution to the SOTA problem, in theory, for most practical routing problems. In this section we present empirical results to validate this claim. Figure 5 shows the actual run-times (in CPU time) for two sample origin-destination (OD) pairs, when computing the optimal policy over a range of travel-time budgets.

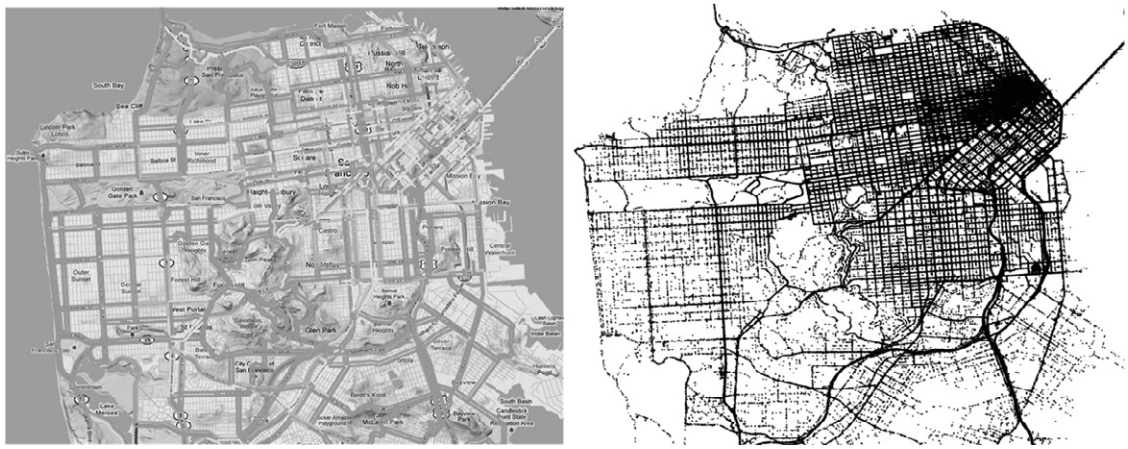


Figure 4: Best viewed in color. **Travel-time estimates on the San Francisco arterial network. Right:** Cumulated probe location measurements collected on July 29th by the *Mobile Millennium* system. **Left:** Travel-time estimates on July 29th at 8 pm; green links correspond to free flow travel-times and red spots correspond to congestion.

As illustrated in Figure 5, the FFT based algorithm with optimal ordering performs significantly better than the brute force approach[5] in both networks (17 minute gain for a 1 hour policy on the highway network and 10 minute gain for a 30 minute policy on the arterial network) and makes the SOTA problem more tractable for real-time applications. The proposed algorithm performs much better on the highway network, where most of the loops have large minimum travel-times and allow the cumulative distribution function $u_i(\cdot)$ to be updated in larger increments (as explained in Section 4.2), making the convolution product more efficient. The FFT algorithm with a random update order performs quite poorly, especially in the case of the arterial network, where it takes approximately 70 minutes to compute a 30 minute policy.[6] This observation agrees with the lower bound in Proposition 5 and the example values in Table 1, since the relative efficiency of the FFT algorithm increases exponentially with the travel-time of the minimum length loop for each node.

5.1.2. Comparison with classical routing algorithms

Most common routing algorithms rely solely on the knowledge of the travel-time as a deterministic quantity when generating optimal route choices. This deterministic travel-time can be inferred for instance from the speed limitations on the network, from historical realized travel-times (e.g. average, worst case), or from a real-time deterministic output of a traffic information system (e.g. mean travel-time, median travel-time).

We compare the performance of the SOTA algorithm to these classical methods on both the highway and arterial networks defined above. First we instantiate the case of a commuter traveling on the highway network from Berkeley (latitude: 37.87201, longitude: −122.3056) to Palo Alto (latitude: 37.4436, longitude: −122.1176), on July 29th, on two departure times, 6:45 am and 8:00 am. This is a typical Bay Area commute experienced by a large population of the San Francisco Bay Area every day. Different optimal routes are possible; for instance the route with minimum expected travel-time (LET route), the route which minimizes the travel-time at the speed limit (speed limit based route), the route with the shortest distance (distance based route), the route which maximizes the probability of arriving on time (SOTA route). We generate optimal routes for all of the above strategies using traffic estimates from the *Mobile Millennium* system. The LET and SOTA routes are computed using the time-dependent implementations of these algorithms. Once the routes are determined, we compute the travel-time distributions for each of these routes

[5] Nie and Fan (2006) show that their discrete convolution algorithm dominates the method given in Fan and Nie (2006) in terms of runtime. We refer the reader to Table 3 in Nie and Fan (2006) for the comparison. Therefore, we compare our algorithm to the algorithm given in Nie and Fan (2006).

[6] The computation time for the FFT algorithm with a random update order is not displayed in Figure 5, since it takes much longer than the other algorithms, to improve the readability of the plot.

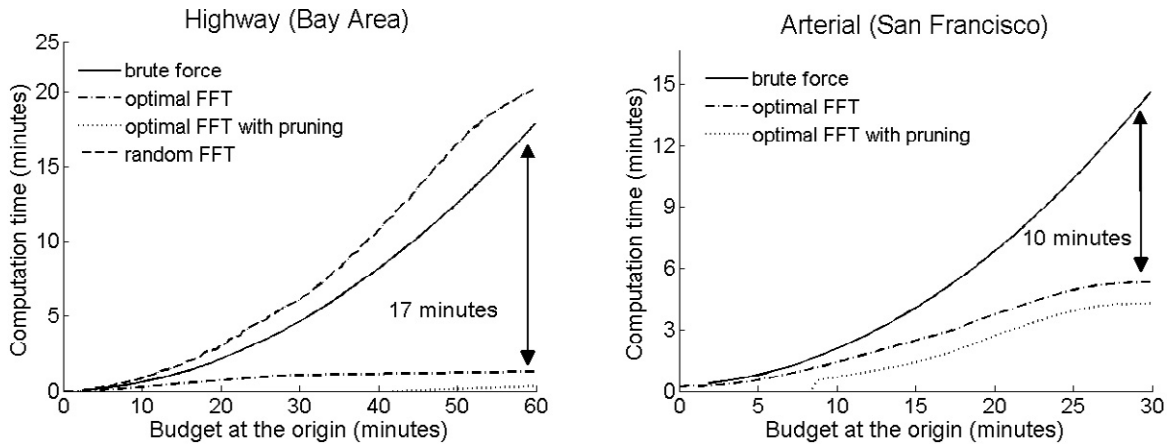


Figure 5: **Illustration of the tractability of the problem. Comparison of run-times (CPU time) for the brute force convolution (solid line), randomly ordered FFT (dashed line), optimally ordered FFT (dot-dash line) and optimally ordered FFT with pruning (dotted line): Left: Highway network** Runtime for computing the optimal policy from Berkeley to Palo Alto. The time discretization (Δt) is 0.5 seconds. **Right: Arterial network** Runtime for computing the optimal policy for a route from the Financial District (Columbus and Kearny) to the Golden Gate Park (Lincoln and 9th). The time discretization (Δt) is 0.2 seconds.

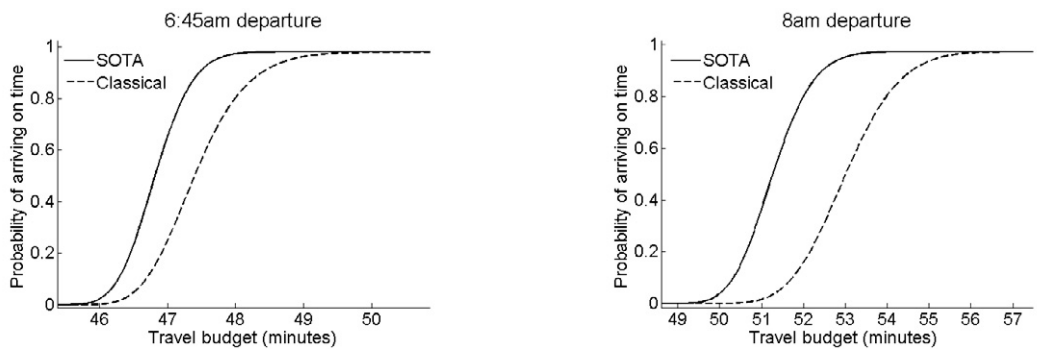


Figure 6: **July** 29^{th}**: Probability of arriving on time at Palo Alto when departing from Berkeley: Left: Departure at** 6**:**45 **am.** The SOTA policy (solid line) provides a higher probability of arriving on time than the choice of the LET route (dashed line). The distance-based route and the speed limit based route are the same as the LET route at this time (Highway I-880). **Right: Departure at** 8**:**00 **am.** The SOTA policy and the LET route provide the same probability of arriving on time (solid line). The speed limit based route and the distance based route (dashed line) are inferior for this criterion.

a posteriori (this is computed by performing a convolution of the individual link travel-time distributions for the links of each route) and determine the probability of arriving within the budget range of 0 to 60 minutes. Figure 6 presents the probability of arriving on time for each of the routing strategies during the budget range.

As traffic conditions vary, the time-dependent SOTA and LET routes change accordingly, while the speed limit based route and the distance based route are static. When departing at 6:45 am, the maximal point wise difference between the SOTA route and the LET route is around 0.4, corresponding to a budget of about 47 minutes. For this budget, the commuter has a 0.65 probability of arriving on time on the SOTA route and a 0.25 probability of arriving on time on the LET route. Naturally, for both the SOTA and LET solutions, the risk of not making the destination on time increases as the budget decreases. However, as illustrated in Figure 6, the SOTA route always provides a higher probability of arriving on time. Furthermore, the SOTA algorithm can provide the user with the probability of on time arrival (i.e. the risk level) for any range of time budgets the user is interested in when the policy is computed, which allows the user to determine whether the risk is acceptable or not and act accordingly.

One may note that the morning congestion build up is visible in Figure 6 since the sharp increase in the cumulative distributions between the left subfigure (6:45 am) and right subfigure (8:00 am) evolves from around 47 minutes to around 52 minutes in this time period. The increase in the area between the SOTA cumulative probability (solid line

from Figure 6) and a second choice route (dashed line from Figure 6) during this time period illustrates that the SOTA policy can dominate classical optimal routes by a higher margin in congestion and non-stationary phases when there is a high uncertainty on the realized travel-time.

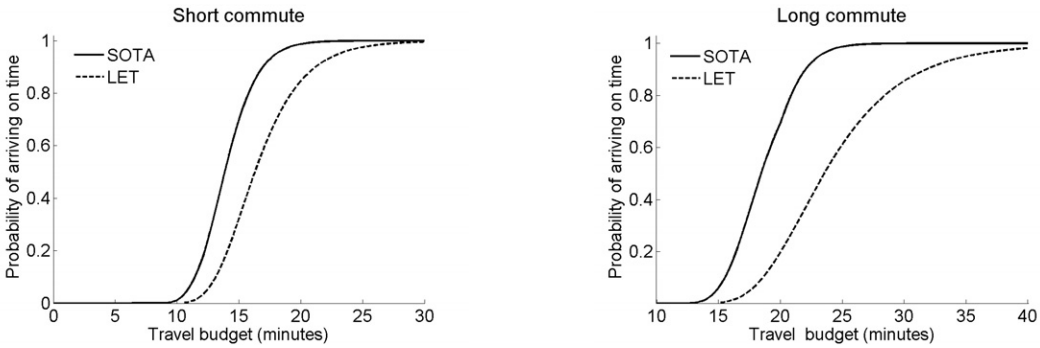


Figure 7: **February 1st: Probability of arriving on time at Left: Fulton and 2nd, and Right: Lincoln and 9th when departing from the Financial District (Columbus and Kearny) at 8:50 pm.** As the graphs imply, the commute to Lincoln and 9th is a longer route than Fulton and 2nd. The relative benefit of using a SOTA policy increases with the route length, since the longer route contains more route choices in the arterial network.

We also compare the performance of the SOTA algorithm on the San Francisco arterial network for two routes starting in the Financial District (Columbus and Kearny) and ending at 1) Lincoln and 9th, and 2) Fulton and 2nd. As seen in Figure 7, the maximal point wise difference between the SOTA route and the LET route for the first example is around 0.4, corresponding to a budget of about 15 minutes, where the commuter has a 0.75 probability of arriving on time on the SOTA route and a 0.35 probability of arriving on time on the LET route. For the second example, the maximal point wise difference is around 0.5, corresponding to a budget of about 22 minutes, where the commuter has a 0.89 probability of arriving on time on the SOTA route and a 0.39 probability of arriving on time on the LET route. The relative benefit of using a SOTA policy increases with the length of a route, since the longer route contains more route choices (with varying cumulative distribution functions) in the arterial network. In the highway network examples from Figure 6, the SOTA policy dominates the LET path for only a 3-5 minute window of the travel-time budget. However, in the arterial network examples, the SOTA policy dominates the LET path for approximately a 10-15 minute window, even though the route lengths were shorter. The reason for this disparity is not limited to these specific examples and is due to the inherent differences of the two networks. The highway network has a limited number of reasonable travel choices from Berkeley to Palo Alto and relatively low variance of the travel-time distributions. Whereas, the arterial network has a large number of route options and highly variable traffic conditions due to the uncertainty introduced by pedestrians, stop signs, traffic lights etc. This results in routes with many distinct cumulative distribution functions and leads to an improved SOTA policy, since the SOTA policy is the upper envelope of all these distinct cumulative distributions functions as illustrated below.

To further illustrate the benefits of the SOTA algorithm when the travel-time distributions are very heterogeneous, we consider the very simple example of two nodes connected by 30 different links each gamma distributed with a mean of 25 minutes, but having different shape and scale parameters. As illustrated in Figure 8, the travel-time distributions for the links are vastly different even though they have the same expected travel-time. A LET routing algorithm could pick any of these links as the optimal solution and in the worst case pick the path that is the worst option for the travel-time budget. On the other hand, the SOTA algorithm picks the best path for a given time-budget, which graphically corresponds to the upper envelope of all the curves. As illustrated by this simple example, the SOTA algorithm has the potential for being relatively more superior to a LET path when the number of travel choices increases and their travel-time distributions are not similar.

5.1.3. Test case: evening rush commute within the city of San Francisco

In this section, we illustrate the adaptive nature of the SOTA algorithm presented in this paper. The output of the algorithm is a policy which accounts for the stochastic nature of link travel-times. Given a budget T, the optimal

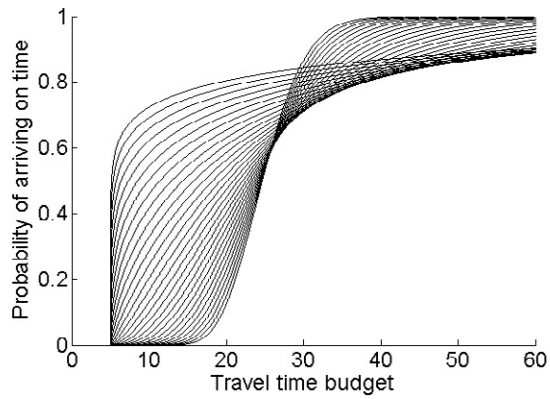


Figure 8: **A family of travel-times distributions modeled using 30 shifted gamma distributions, each with the same mean travel time of 25 minutes and a minimum travel time of 5 minutes.** The shape parameter of the distributions ranges from 4 to 0.13 and the scale parameter of the distributions ranges from 5 to 150. The SOTA policy will be the upper envelope of all the curves. The LET path could be any of the curves and in the worst case even be the path that minimizes the probability of arriving on time for a given budget.

Remaining budget b at point C	Turn direction from East
$b \leq 12$ minutes	Take a left turn
$b \geq 12$ minutes	Continue straight

Table 5: The optimal policy at point C routes on different paths depending on the value of the remaining travel budget with respect to 12 minutes. The probability of arriving on time at B when remaining budget at C is 12 minutes is 0.56.

policy computation encompasses the design of a decision process at each possible intersection of the network; the choice of the optimal route to take from this node depends on the remaining budget.

Here we consider two drivers commuting from point A to point B (see Figure 9) on February 1st. They depart from point A at 8:50 pm and desire to reach point B before 9:10 pm; i.e. their travel budget is 20 minutes. We assume that both drivers are equipped with a mobile device on which the output of the SOTA algorithm is available. At each intersection, they follow the turn directions given by the optimal policy. For this test case we sample the drivers travel-time from the output of the real-time arterial traffic estimation module Herring et al. (2010) from the *Mobile Millennium* system.

Because of different driving behaviors, external factors, link travel-time stochasticity, both drivers will experience different travel-times during their commute. The strength of the SOTA algorithm is that the optimal route choice given by the algorithm is given at every intersection in function of the remaining budget. As illustrated in Table 5, the optimal route to take from point C includes a left turn for low values of the remaining budget. Because the second driver experienced a larger travel-time on the path from point A to point B, he is advised to take a left turn and to follow a path with more variability, and thus higher risk, which may be more appropriate to his situation. The first driver continues straight at point C.

6. Conclusions

This article considers the reliable routing problem of maximizing the probability of on time arrival, which is a computationally difficult problem, and presents algorithmic methods that improve the tractability of the problem over existing methods. First, we prove the existence of a single iteration convergence algorithm to the continuous time version of the problem. The update property of this solution in conjunction with an optimal ordering process and the use of the Fast Fourier Transform is then used to construct an efficient algorithm for computing a discrete approximation to the problem. The correctness of this algorithm when extended to time-varying and correlated link travel-times is also shown. Finally, the theoretical results are validated by implementing the proposed algorithm in

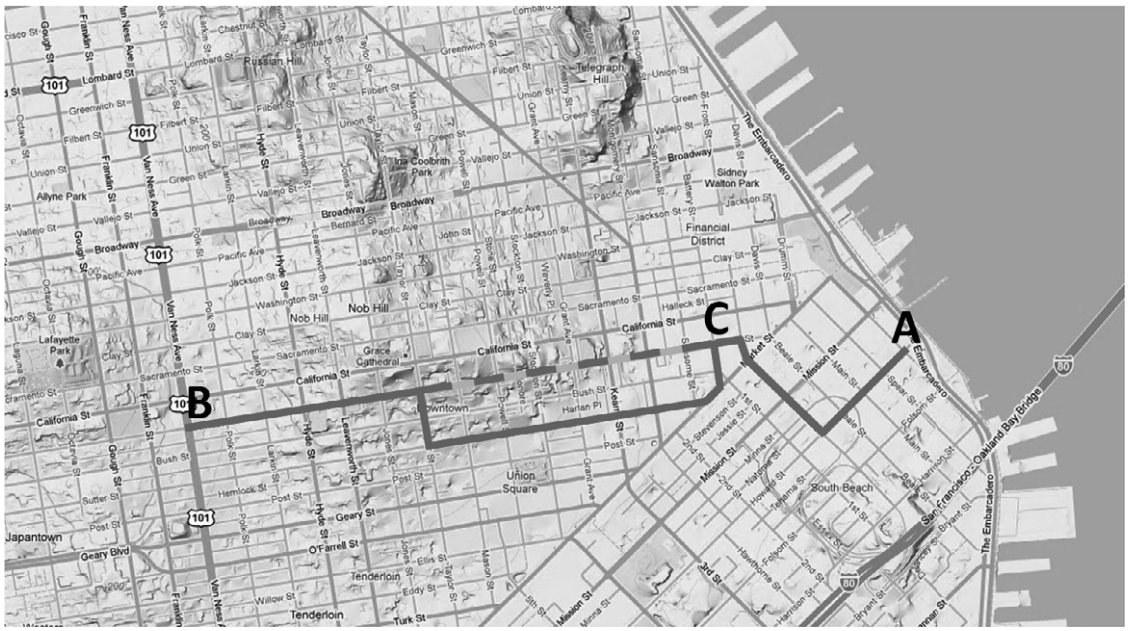


Figure 9: **Commute from point A to point B:** two drivers depart from point *A* at 8:50 pm on February 1st with a budget of 20 minutes to reach point *B*. They are routed by the SOTA module. Because their realized travel times differ, their recommended routes differ. The first driver is suggested to turn left at point *C*, whereas the second driver is suggested to drive straight.

the *Mobile Millennium* traffic information system. Numerical results show that the proposed algorithm provides a significant reduction in the computation time over existing methods, especially in highway networks. Our goal is to provide the theoretical basis for a tractable implementation of adaptive routing with reliability guarantees in an operational setting.

7. Acknowledgments

The authors would like to thank Olivier Goldschmidt, Olli-Pekka Tossavainen, Pierre-Emmanuel Mazaré, Satish Rao, Timothy Hunter and the systems team at the California Center for Innovative Transportation for their contributions. We also gratefully acknowledge the financial support of Telenav and its partnership with the University of California Berkeley.

References

Abraham, I., Fiat, A., Goldberg, A., Werneck, R., 2010. Highway dimension, shortest paths, and provably efficient algorithms. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA10), Society for Industrial and Applied Mathematics.

Assaf, D., Levikson, B., 1982. Closure of phase type distributions under operations arising in reliability theory. The Annals of Probability 10 (1), 265–269.

Astarita, V., 1996. A continuous time link model for dynamic network loading based on travel time function. In: 13th International Symposium on Transportation and Traffic Theory. Lyon, France, pp. 79–102.

Aubin, J., 2001. Viability Theory. Springer.

Bellman, R., Kalaba, R., 1966. Numerical Inversion of the Laplace Transform. American Elsevier Publishing Company.

Bertsekas, D., 2005. Dynamic Programming and Optimal Control. Athena Scientific.

Bertsekas, D., Tsitsiklis, J., 1996. Neuro-dynamic Programming. Athena Scientific.

Block, H., Savits, T., 1976. The ifra closure problem. The Annals of Probability 4 (6), 1030–1032.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., 2001. Introduction to Algorithms. The MIT Press.

Dean, B., 2004. Algorithms for minimum-cost paths in time-dependent networks with waiting policies. Networks 44, 41–46.

Dijkstra, E., 1959. A note on two problems on connection with graphs. Numerishe Mathematik 1, 269–271.

Dreyfus, S., 1969. An appraisal of some shortest-path algorithms. Operations Research 17, 395–412.

Fan, Y., Kalaba, R., Moore, J., 2005. Arriving on time. Journal of Optimization Theory and Applications 127 (3), 497–513.
Fan, Y., Nie, Y., 2006. Optimal routing for maximizing travel time reliability. Networks and Spatial Economics 3 (6), 333–344.
Frank, H., 1969. Shortest paths in probabilistic graphs. Operations Research 17, 583–599.
Fu, L., Rilett, L., 1998. Expected shortest paths in dynamic and stochastic traffic networks. Transportation Research Part B 32 (7), 499–516.
Geisberger, R., Sanders, P., Schultes, D., Delling, D., 2008. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In: Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08). Springer, pp. 319–333.
Goldberg, A. V., Kaplan, H., Werneck, R., 2007. Better landmarks within reach. In: Workshop on Experimental Algorithms (WEA), Rome, Italy.
Hall, R., 1986. The fastest path through a network with random time-dependent travel times. Transportation Science 20 (3), 182.
Herring, R., Hofleitner, A., Bayen, A., 2010. Estimating arterial traffic conditions using sparse probe data. In: 13th International Conference on Intelligent Transportation Systems. Madeira Island, Portugal.
L'Ecuyer, P., 2008. Stochastic simulation in java, http://www.iro.umontreal.ca/ simardr/ssj/indexe.html.
Loui, R., 1983. Optimal paths in graphs with stochastic or multidimensional weights. Communications of the ACM. 26 (9), 670–676.
Miller-Hooks, E., Mahmassani, H., 2000. Least expected time paths in stochastic, time-varying transportation networks. Transportation Science 34 (2), 198–215.
Nie, Y., Fan, Y., 2006. Arriving-on-time problem. Transportation Research Record, 193–200.
Orda, A., Rom, R., 1990. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. Journal of the ACM 37 (3), 607–625.
Sedgewick, R., 1990. Algorithms in C. Addison Wesley Publishing Company.
Mobile Millennium, 2008. http://traffic.berkeley.edu.
Waller, S., Ziliaskopoulos, A., 2002. On the online shortest path problem with limited arc cost dependencies. Networks 40 (4), 216–227.
Wendykier, P., 2009. Jtransforms, http://sites.google.com/site/piotrwendykier/software/jtransforms.
Work, D., Blandin, S., Tossavainen, O.-P., Piccoli, B., Bayen, A., 2010. A traffic model for velocity data assimilation. AMRX Applied Mathematics Research eXpress 1, 1–35.

Appendix A

Proposition 5. *The travel budget* $t \leq \Delta t\, 2^{\frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)} - \delta$ *is a lower bound for when the FFT based approach has a faster runtime that the brute force approach.*

Proof of proposition 5.

We compare the runtime of the FFT approach and the brute force approach. In order to compute the optimal solution up to a given time τ, the brute force method needs to be executed for $\frac{\tau}{\Delta t}$ steps and the FFT method needs to be executed for $\frac{\tau}{\delta}$ steps. The runtime of the brute force method is $\sum_{k=1}^{\frac{\tau}{\Delta t}} k$ and the running time of the FFT approach is $\sum_{k=1}^{\frac{\tau}{\delta}} \frac{\delta k}{\Delta t} \log(\frac{\delta k}{\Delta t})$. The FFT approach is faster than the brute force convolution for $t = K\,\delta$ such that:

$$\sum_{k=1}^{K} \frac{\delta k}{\Delta t} \log \frac{\delta k}{\Delta t} \leq \sum_{k=1}^{\frac{K\delta}{\Delta t}} k = \frac{1}{2} \frac{K\delta}{\Delta t} \left(\frac{K\delta}{\Delta t} + 1 \right).$$

If we use the fact that the function $x \mapsto x \log x$ is increasing on $[1; +\infty)$, we can bound the left hand side of the above inequality as follows:

$$\begin{aligned} \sum_{k=1}^{K} \frac{\delta k}{\Delta t} \log \frac{\delta k}{\Delta t} &< \int_{z=0}^{K} \frac{\delta(z+1)}{\Delta t} \log \frac{\delta(z+1)}{\Delta t}\, dz \\ &= \frac{1}{2} \frac{\delta}{\Delta t} (K+1)^2 \left(\log \frac{\delta\,(K+1)}{\Delta t} - \frac{1}{2} \right) - \frac{1}{2} \frac{\delta}{\Delta t} \left(\log \frac{\delta}{\Delta t} - \frac{1}{2} \right) \end{aligned}$$

where the inequality is a right Riemann integral bound. A lower bound t on the time up to which the FFT approach is faster than the brute force convolution thus satisfies:

$$\frac{1}{2} \frac{\delta}{\Delta t} (K+1)^2 \left(\log \frac{\delta\,(K+1)}{\Delta t} - \frac{1}{2} \right) - \frac{1}{2} \frac{\delta}{\Delta t} \left(\log \frac{\delta}{\Delta t} - \frac{1}{2} \right) \leq \frac{1}{2} \frac{K\delta}{\Delta t} \left(\frac{K\delta}{\Delta t} + 1 \right).$$

If we assume $\delta \geq \Delta t \exp \frac{1}{2}$, we have $-\frac{1}{2}\frac{\delta}{\Delta t}\left(\log \frac{\delta}{\Delta t} - \frac{1}{2}\right) \leq 0$ and thus a sufficient condition to have the inequality above satisfied is to have:

$$\frac{1}{2}\frac{\delta}{\Delta t}(K+1)^2\left(\log\frac{\delta(K+1)}{\Delta t} - \frac{1}{2}\right) \leq \frac{1}{2}\frac{K\delta}{\Delta t}\left(\frac{K\delta}{\Delta t}+1\right)$$

We can equivalently rewrite the above inequality as:

$$(K+1)^2 \log\frac{\delta(K+1)}{\Delta t} \leq \frac{1}{2}(K+1)^2 + \frac{\delta}{\Delta t}K^2 + K.$$

We wish to find $\alpha \in \mathbb{R}$ such that $\alpha(K+1)^2 \leq \frac{1}{2}(K+1)^2 + \frac{\delta}{\Delta t}K^2 + K$ for $K \geq 1$. This is satisfied for $\alpha \leq \frac{1}{4}\left(3 + \frac{\delta}{\Delta t}\right)$, which allows us to write that the FFT is faster than the brute force approach when:

$$(K+1)^2 \log\frac{\delta(K+1)}{\Delta t} \leq \frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)(K+1)^2$$

which is equivalent to:

$$K \leq \frac{\Delta t}{\delta} 2^{\frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)} - 1$$

and thus a lower bound t reads:

$$t \leq \Delta t\, 2^{\frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)} - \delta$$

Appendix B

Proposition 7. *The ordering that gives the optimal solution to the optimization problem (9) can be obtained using Algorithm 4 in $O(\frac{mT}{\Delta t}\log(n))$ time, where n and m are respectively the number of nodes and links in the network, Δt is the time discretization interval and T is the time budget.*

Proof of proposition 7.

Algorithm 4 begins at the termination condition of the SOTA problem, the source node being updated to the budget, and recursively builds (in reverse order) the optimal sequence of updates that allow the source node to be updated to the budget. Thus, the algorithm is initialized with the terminal condition of $\tau_r = T$. This is the initial constraint of the optimal ordering algorithm. Reaching this condition must be preceded by all the downstream nodes $j \in \pi_r$ of the source node r being updated to at least $\tau_r - \delta_{rj}$, since the correctness of the algorithm requires the invariant in equation (8),

$$\tau_i \leq \min_j(\tau_j + \delta_{ij}) \;\; \forall (i,j) \in A$$

to hold. Therefore, the initial constraint $\tau_r = T$ is relaxed by adding these new constraints to the constraint list ψ. At the same time, we also add $\tau_r = T$ to the optimal order stack χ. This will be the final update in the optimal ordering, since we are building the list from the last update to the first.

Once we have a set of new constraints, we need to decide which node to relax and how far to update τ_i. The contribution from a given node i to the objective function of the optimization problem (9) is minimized when the τ_i value for that node is minimized as much as possible. Furthermore, lowering the τ_i value for a node reduces the new constraints introduced when relaxing that node. Therefore, an optimal update should reduce the τ_i value of a node as much as possible such that invariant (8) is not violated.

The next step is to determine which constraint from the constraint list ψ to relax first. We need to show that that the order of relaxation guarantees that the algorithm will not introduce any new constraints that violate any updates done in previous relaxations. Picking the node i with the largest constraint τ_i in ψ guarantees this, since δ_{ij} is strictly positive and the new constraints $\tau_j\,(\forall j \in \pi_i)$ that are added satisfy the condition $\tau_j \leq \tau_i - \delta_{ij}$, which implies that node i cannot have a new constraint that is greater than its current constraint τ_i at any future point of the algorithm. Therefore, correctness is preserved by relaxing the node i with the largest constraint τ_i in ψ and setting its value to $\tau_i - \delta_{ij}$. Node i is then added to the optimal order stack χ. The new constraints introduced by setting node i to this value are then added to ψ, if $\tau_i - \delta_{ij} > 0$ and $j \neq s$. It is unnecessary to add constraints if these conditions are not

satisfied, since $u_i(t) = 0\,(\forall\, i \in N, t \leq 0)$ and $u_s(t) = 1\,(\forall\, t \geq 0)$. This process is performed recursively until the list ψ is empty. The process is guaranteed to terminate because the values of new constraints that are added when relaxing an existing constraint are monotonically decreasing.

The complexity of Algorithm 4 is $O(\frac{mT}{\Delta t}\log(n))$. The *Extract* and *Insert* operations of the Algorithm 4 can be replaced with a single *IncreaseKey* operation, which runs in $O(log(n))$ time (see Sedgewick (1990); Cormen et al. (2001) for details). The *IncreaseKey* operation will increase the key of a given node if the new key is greater than its existing value. This is exactly what the *Extract* and *Insert* operations are used for. The pseudo-code for Algorithm 4 uses the *Extract* and *Insert* operations to improve readability. The *ExtractMax* operation can be performed in constant time. Therefore, each iteration of the algorithm takes $O(log(n))$ time. In the worst case, each link might need to be updated $\frac{T}{\Delta t}$ times. Repeating this over the m links of the network, we obtain a complexity of $O(\frac{mT}{\Delta t}\log(n))$. ■